Effect of Native Language on Student Learning and Classroom Interaction in an Operating Systems Course

Adalbert Gerald Soosai Raj
Department of Computer Sciences and Education
University of Wisconsin-Madison
gerald@cs.wisc.edu

Eda Zhang
Department of Curriculum and Instruction
University of Wisconsin-Madison
rzhang346@wisc.edu

Saswati Mukerjee
Department of Information Science and Technology
College of Engineering Guindy, Anna University
msaswati@auist.net

Jim Williams
Department of Computer Sciences
University of Wisconsin-Madison
jimw@cs.wisc.edu

Richard Halverson
Department of Educational Leadership and Policy Analysis
University of Wisconsin-Madison
rich.halverson@wisc.edu

Jignesh M. Patel
Department of Computer Sciences
University of Wisconsin-Madison
jignesh@cs.wisc.edu

ABSTRACT

Understanding an operating systems (OS) code base is a difficult task since it involves understanding a huge amount of low-level C and assembly code. The inherent level of difficulty associated with OS topics is high because of the high element interactivity (i.e., material consists of elements that heavily interact). The mental effort associated with learning a complex subject like OS may be higher for non-native English speakers, when the subject is taught in a natural language (i.e., English) that is not the students' native language. We were interested in finding the effect of an instructional design that combines the students' native language along with English on students' understanding of select topics in OS. We designed an experiment to teach CPU virtualization using xv6 to two groups of undergraduate students in Tamil Nadu, India. We taught the experimental group using English and Tamil (native language of students in Tamil Nadu) and the control group using only English. We conducted a pre-test and a post-test to test students' understanding of the OS topics taught, before and after our intervention respectively. We also collected data on the questions that students asked in lectures during our intervention. We found that teaching OS using native language and English is no different than teaching OS using only English with respect to student learning. We also found that the native language had an impact on the student engagement and classroom interaction by creating more dialogue within the Tamil+English (experimental) classroom when compared to the English-only (control) classroom.

ITiCSE '19, July 15–17, 2019, Aberdeen, Scotland, UK

ACM ISBN 978-1-4503-6301-3/19/07…$15.00
https://doi.org/10.1145/3304221.3319787

CCS CONCEPTS

• **Social and professional topics → Computer science education**;

KEYWORDS

Bilingual CS Education, Operating Systems, Native Language

ACM Reference format:
Adalbert Gerald Soosai Raj, Eda Zhang, Saswati Mukerjee, Jim Williams, Richard Halverson, and Jignesh M. Patel. 2019. Effect of Native Language on Student Learning and Classroom Interaction in an Operating Systems Course. In *Proceedings of Innovation and Technology in Computer Science Education, Aberdeen, Scotland, UK, July 15–17, 2019 (ITiCSE '19),* 7 pages.
https://doi.org/10.1145/3304221.3319787

1 INTRODUCTION

Students in India learn their subjects during K-12 (kindergarten (K) through twelfth grade (12)) either in English or their native language (i.e., the language that a person has spoken from earliest childhood). This choice of language depends on whether the student studies in an English-medium school or a native-language-medium school during their K-12. The decision about which medium of instruction a child studies is made by the child's parents. The parents make these decisions mainly based on their financial status. Tamil-medium education is free of cost but English-medium education costs some money. This difference in costs forces the students from poor financial backgrounds to learn only in Tamil-medium as these students' parents are not in a position to afford an English-medium school for their children's education.

Although there are two mediums of instruction during K-12, Computer Science is taught in undergraduate institutions primarily in English. The main reason for teaching CS in English is because English is the de facto international language of programming [9]. Therefore, students who aren't very comfortable in English (e.g., students who studied in a Tamil-medium school during their K-12 and students who aren't proficient with English even though they may have studied in an English medium school) find it difficult to

understand introductory programming concepts since the subject is already difficult and they are also forced to learn it in a language that they are not comfortable with [13]. As a result, these students end up failing their CS courses, and eventually develop inferiority complexes about their abilities [12].

Most of the prior work in the area of bilingual CS education have been focused mainly on teaching introductory programming [13, 20]. Although the upper-level CS courses like the operating systems are considered to be inherently more difficult than the intro programming courses, to the best of our knowledge, no previous work has been done in studying the effect of a students' native language in learning advanced subjects like the operating systems. We consider our work as a first small step in this direction.

Our research tries to answer the following two questions:

(1) *What is the effect of using the native language of non-native English speakers on student understanding of concepts/code in an upper level Computer Science course, like introduction to operating systems (using xv6 codebase)?*
(2) *What is the effect of native language on classroom interaction measured using the number and the quality of questions asked during a class?*

To answer these questions and to better understand the impact of the medium of instruction to teach upper-level CS classes, we conducted an experiment where we taught an OS course based on xv6 [4] (a simple unix-like teaching operating system) using both *Tamil and English* to a group of students (experimental group) whose native language was Tamil. We also taught the same OS course to another set of students (control group) *only in English*, even though their native language was also Tamil. We used *Tamil along with English* to teach the experimental group since we believe that even though Tamil may help students to better understand programming concepts, they should learn English too since English is the global language for programming and communication [11].

2 RELATED WORK

Soosai Raj et. al. [20] studied the effect of native language on student learning and sentiments and found that teaching linked lists in data structures using Tamil and English was no different than teaching using only English. They also found that students have expressed positive sentiments about learning using two languages, based on the student feedback data [19]. We consider our study to be an extension of this work applied to the domain of an upper-level CS class.

John Airey [1] studied undergraduate physics classes that were taught in English and Swedish to 22 students at two Swedish universities. The lectures were recorded and students were interviewed using stimulated recall. He found that when taught in English the students asked and answered fewer questions and reported being less able to simultaneously follow the lecture and take notes. Students adapted to being taught in English by; asking questions after the lecture, no longer taking notes in class, reading sections of work before class or - in the worst case - by using the lecture for mechanical note taking. Our study is an application of this work in the context of computer science education. We made classroom observations and made a note of the classroom interactions based on the questions asked by the students.

Pal and Iyer [14–16] studied the effect of using native language (Hindi) on student learning for non-native English speakers. They found that Hindi-medium students understood programming concepts better (measured using post-tests), when they were taught using Hindi compared to being taught using only-English. Our study differs from this study in the medium of instruction used for teaching programming: they used a Hindi-only approach while we used a bilingual (Tamil+English) approach.

Probyn interviewed some teachers in South Africa who use English along with Xhosa, an official South African language, to teach Science [17, 18]. The study shows that the language of learning and teaching frequently creates a barrier to learning when it is not the native language of the learners. The benefits of code-switching between the two languages for increased comprehension among students is also highlighted. We consider our study as an extension to this study where we try to find if the vernacular has any effect on learning programming, mainly by measuring student learning using pre-and-post technical tests.

3 METHODOLOGY

In this section, we explain the methodology that we used to conduct our experiment and collect our data.

3.1 Participants

The experiment was conducted in a well reputed Engineering college in Tamil Nadu. Two groups of third-year students, enrolled in two different sections of an operating systems course were selected as participants for our study. One group was treated as the control group and the other group was treated as the experimental group. The total number of students in the control and experimental group were 51 and 39 respectively. All these students have previously taken a programming and data structures course in C [10]. They have also taken an introductory course on the basic concepts of an operating systems (e.g., scheduling policies, page replacement policies, process creation APIs, concurrency using threads, etc) in which they learn policies and APIs of an OS. The course in which we conducted our intervention was the next course that students take after the basic OS course in which they learn the mechanisms and implementation details of CPU/memory virtualization, and File Systems in a unix-like OS.

3.2 Experimental Design

The experiment was conducted using a *nonrandomized control group pre-test post-test design* [5]. In this design, the participants were not randomly assigned to groups but remained in their pre-assigned groups. This was done as we conducted our experiments as part of regular classes and so were not able to randomize and regroup our participants into two new groups. This experimental design helped us to keep the details of our experimental procedure to be a secret from the students until the end of our intervention. This increases the external validity of the design by reducing the reactive effects of the experimental procedure [5]. We acknowledge that the problem with this approach is that even if there are any post-test differences between the groups, they may be attributed to characteristic difference between the groups rather than to the intervention. We take this issue into consideration in our result

analysis by choosing our statistical models for pre-test post-test comparison very carefully (see Section 4 for more details). The teacher, who was the actual instructor for the two sections of this course, agreed to teach the two groups during our intervention.

The teachers introduced the researcher as a visiting faculty who will observe their classes and co-teach a few classes whenever needed. The researcher was a silent observer who took field notes during the intervention. The researcher helped the teacher answer a few questions on xv6 [4] codebase whenever the teacher needed some help. It is important to note that, as a part of our experiment, xv6 was introduced for the first time in this operating systems course, as a way to explain the concepts underlying the unix operating systems. As xv6 was a new codebase even for the instructor, the researcher who was familiar with the codebase before, helped the teacher in answering a few questions that the students asked about the codebase during the lectures.

3.3 Experimental Procedure

The following activities were performed with both the English-only (control) group and the Tamil+English (experimental) group as a part of our intervention. There was a pre-test, in-class lectures, code walkthrough sessions, and a post-test. The questions in the pre-test and the post-test were in English for both groups.

3.3.1 Pre-test. A pre-test was conducted to determine the students' understanding of the basic concepts of an OS. There were four questions on the pre-test. There was one question each on CPU scheduling policies, page replacement algorithms, creating new processes using fork and exec, and concurrency using threads. The pre-test was conducted for a total of 24 points, six points for each question. The pre-test questions were created based on the previous topics that students learned in the basic OS course, in consultation with the instructor who taught that course.

The above mentioned topics were tested in the pre-test as they would give us a baseline for understanding the students' prior knowledge about an OS before our intervention. The complete pre-test can be found at this link: http://bit.do/os_pretest.

3.3.2 Classroom Lectures. Twelve classroom-based lectures, each of 50 minutes duration, were presented for both groups. The mechanisms behind CPU virtualization were explained in those twelve lectures. Topics discussed were: process creation, limited direct execution, system calls, timer interrupt, context switch, etc. The same topics were taught to both the groups. The free online textbook, Operating Systems: Three Easy Pieces (OSTEP) [3], was used as a reference to teach these topics.

The main differences between the lectures for the two groups were the following: The lectures were taught *only in English* for the students in the control group. Also, the students in the control group were required to communicate with the instructor and their classmates during the lecture in English only.

On the other hand, the lectures were taught using both *English and Tamil* in the experimental group, and the students were free to communicate in any of those two languages, whichever they felt more comfortable with. The instructor used both English and Tamil nearly equally (i.e., 50% time in English - 50% time in Tamil) while teaching the experimental group. The instructor answered the questions during the lecture using the same language (either English or Tamil) in which they were asked.

The instructor used bilingual teaching methods like code-switching [6] and translanguaging [7] for teaching the Tamil+English (experimental) group. The instructor used *code-switching* for switching between Tamil and English in the following way. She used English to introduce a topic, to explain the basic idea behind the topic, and to explain some technical terms (e.g., context switch). She switched to Tamil whenever she felt that a particular topic needed detailed explanation in order to help the students understand the idea in a better way (e.g., How does a context switch happen?). The instructor used *translanguaging* as follows. She used Tamil only for oral explanations, discussion, and answering students' questions. She wrote all the content (e.g., topic names, definitions, explanations etc.) on the chalk-board during the lectures only in English.

3.3.3 Code Walkthrough Sessions. The instructor conducted regular code walkthrough sessions as a part of our intervention, where she helped students understand the code flow of the OS topics discussed in the lectures (e.g., system call, scheduler, context switch). xv6 code was used during these code walkthrough sessions for understanding the internal workings of a unix-like operating system. The code walkthrough sessions were conducted as a part of the twelve lectures (see Section 3.3.2). Approximately, six of the twelve classroom lectures were used for code walkthroughs and the remaining six were used for explaining OS concepts about CPU virtualization.

The only difference between the code walkthrough sessions for the two groups was the language that was used while explaining the xv6 code: Tamil and English was used for the code explanations in the experimental group while English only was used for code explanations in the control group.

3.3.4 Post-test. The post-test consisted of 7 questions on CPU virtualization. The post-test questions were open-ended and based on the following topics: (1) Process state transitions, (2) Trapping into the kernel, (3) System calls vs library functions, (4) Process Control Block (PCB) in xv6, (5) User vs kernel threads, (6) `syscall()` function in xv6, and (7) Steps in servicing a system call.

All the questions on the post-test were based on the material taught during the classroom-based lectures and the code walkthrough sessions. The post-test was conducted for a total of 25 points. The complete post-test can be found at this link: http://bit.do/os_posttest.

3.3.5 Classroom observations. The researcher observed two regular classes before our intervention. Classes were taught only in English to both the groups before our intervention. Our intervention was conducted only for 6 weeks during the regular semester (which is usually 15 weeks long).

During our intervention, the researcher acted as a silent observer and took field notes of the questions asked by the students, answers given by the instructor, and the follow-up questions that were arised and answered. The observer also made notes of the language used for asking the questions. One sample interaction in the experimental group was as follows (Q - question asked by the student; A - answer given by the teacher): *Q: Why do we save the context of process A twice during a context switch? A: We save it twice because if we are*

switching to a different process, then we need to save the context of both the user thread and the kernel thread of process A. Q: So, do we use different memory locations to store these two different contexts? A: Yes, we do. The context of the user thread is stored in a structure called a trap frame and the context of the kernel thread is stored in a structure called process control block in xv6. The complete set of questions asked by students in both the groups and the answers given by the instructor can be found here: http://bit.do/os_questions

4 RESULTS

The mean of the pre-test scores and the post-test scores for the two groups are shown in Table 1 and Table 2 respectively.

Table 1: Mean of pre-test scores for the two groups

Group	N	Mean	Std. Dev.	Std. Error of Mean
Control	51	48.0	16.9	2.4
Experimental	39	58.2	14.7	2.4

Table 2: Mean of post-test scores for the two groups

Group	N	Mean	Std. Dev.	Std. Error of Mean
Control	51	55.1	20.2	2.8
Experimental	39	59.7	20.4	3.3

4.1 Analysis of Pre-test Scores

We compared the pre-test of both groups using an independent samples t-test. The following assumptions for two sample t-test were satisfied:
(1) The Central Limit Theorem (CLT) applies to each sample individually. i.e., pre-test data is normally distributed in both the groups as confirmed by the Quantile-Quantile (Q-Q) plots.
(2) The pre-test scores of both the groups are random samples that are independent of each other.
(3) The variances of the pre-test scores are approximately equal. (i.e., Standard deviation of control group (s1) = 16.9; Standard deviation of experimental group (s2) = 14.7; s1/s2 = 1.1 < 1.5).

We performed an independent samples t-test to compare the pre-test scores between the control group and the experimental group and found *a significant difference* in pre-test scores between the two groups ($t(90) = -3.05$, $df = 88$, $p = 0.003$). This means that the two groups differed significantly with respect to their prior knowledge on OS concepts. Therefore, we cannot directly compare the post-test scores between the two groups using independent samples t-test [5]. Instead we intend to do the following two types of analysis:
(1) Compare the gain scores (post-test - pre-test) of the two groups using an independent samples t-test (if t-test assumptions are satisfied).
(2) Compare the post-test scores of sub-groups of students having the same pre-test score between the two groups using ANalysis of COVAriance (ANCOVA) [8]. Note that post-test analysis was not possible since the assumptions for ANCOVA was not satisfied (see Section 4.3).

4.2 Analysis of Gain Scores

The mean of the gain scores for the two groups are shown in Table 3. The mean gain for the control group is higher than that of the experimental group. The assumptions for independent samples t-test (as shown in Section 4.1) were satisfied by the gain scores of both the groups. We performed independent samples t-test [8] to compare the gain scores between the control group and the experimental group and found *no significant difference* in gain scores ($t(90) = 1.25$, $df = 88$, $p = 0.21$).

Table 3: Mean of gain scores for the two groups

Group	N	Mean	Std. Dev.	Std. Error of Mean
Control	51	7.1	18.9	2.6
Experimental	39	1.5	22.7	3.6

4.3 Analysis of Post-test Scores

We intended to use ANCOVA to find if there was any difference in the post-test score (response variable) due to the students' group (independent variable), while controlling for the students' pre-test scores (covariate). All assumptions of ANCOVA were satisfied for our data, except the homogeneity of regression lines. Hence, we were not able to compare the post-test differences between the two groups using ANCOVA as explained below.

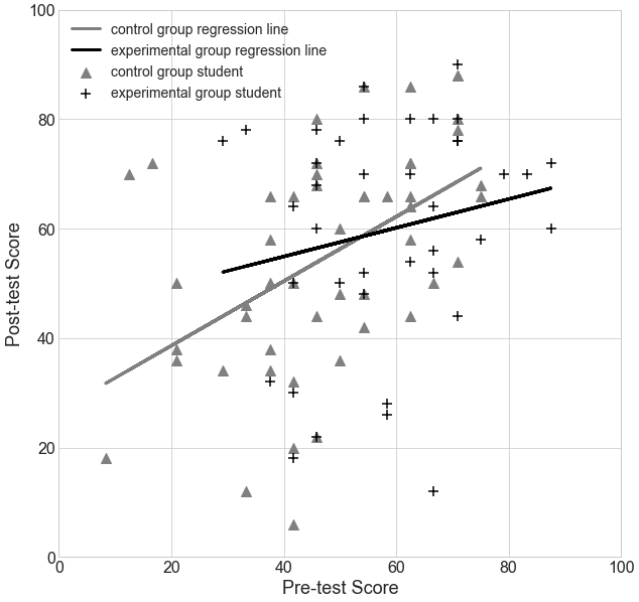

Figure 1: A scatter plot showing each student's pre-test score plotted on the x-axis and the post-test score plotted on the y-axis. The regression lines are also plotted for each group which shows the relationship between the pre-test and post-test scores.

We plot a scatter plot in Figure 1 using the pre-test and the post-test scores of the students in both the groups. In this scatter plot, the x-axis represents the pre-test scores of the students and the y-axis represents the post-test scores of the students. The two lines shown in this scatter plot are the regression lines for a particular group that summarizes the relationship between the post-test score and the pre-test score for that group.

The regression lines for both the groups have *positive slopes* (0.26 for the experimental group and 0.59 for the control group) and the regression lines are above the diagonal (i.e., the line connecting the points (0,0) and (100,100)), which means that our intervention had a similar positive effect on both the groups. But the slopes are not nearly equal since the slope of the regression line for the control group (0.59) is more than twice the slope of the regression line for the experimental group (0.26). This means that we cannot apply ANCOVA to find if there is any significant difference between the post-test scores of the two groups.

4.4 Analysis of Student Questions

The number of questions asked by the students in both the groups is shown in Table 4. The total number of questions asked by the students in the Tamil+English (experimental) group (# questions = 11) is nearly three times more than the number of questions asked by the students in the English-only (control) group (# questions = 4).

Table 4: Number of questions asked by the students in the two groups

Group	Total questions	# English	# Tamil
Control	4	4	0
Experimental	11	2	8

In the experimental group, the number of questions asked in Tamil (# questions = 8) were more than the number of questions asked in English (# questions = 2) by a factor of 4 (i.e., a ratio of 4:1 in favour of Tamil). One student asked a question in English and followed up with another question in Tamil. This student's question was not considered to be in either of the two groups and hence the number of questions in English (2) + number of questions in Tamil (8) is one less than the total number of questions (11). In the control group, there weren't any question that were asked in Tamil since the students were required to communicate only in English.

One interesting observation was that, even though there were not as many questions in the control group as there were in the experimental group during the lectures, the number of students who asked questions on a one-on-one basis after class was more in the English-only (control) group than in the Tamil and English (experimental) group.

We classify the questions asked by the students in both the groups into the following four types of knowledge dimensions based on revised Bloom's taxonomy [2]:
(1) Factual knowledge (e.g., What is an address space?)
(2) Conceptual knowledge (e.g., Why doesn't a program have stack or heap when it is on disk?)
(3) Procedural knowledge (e.g., When we switch from one process to another are there really two context switches that happens?)
(4) Metacognitive knowledge (e.g., Why do we save the context of process A twice during a context switch?)

A few questions were classified under more than one knowledge domain. For example, the following question asked by a student started as a factual question but the follow-up question was in the conceptual knowledge domain (Q - Question; A - Answer): *Q (Student): What is static data? A (Teacher): Static data is the part of the address space where global and static variables that are common for all functions are stored. Q (Student): Why can't those variables be stored on the stack? A (Teacher): Good question. They can't be stored on the stack since these variables should be accessible from any function. A stack frame for a function only stores the local variables that were created within the function and hence these local variables are accessible only from within the function that is currently executing.* Note that even though there were two questions that were asked as a part of this interaction, it was treated as only one question as follow-up questions were considered to be part of the original question.

The number of questions that are classified across the four different knowledge dimensions is shown in Table 5.

Table 5: Classification of questions across 4 knowledge domains

Group	Factual	Conceptual	Procedural	Metacog.
Ctrl	3	0	1	0
Exp	4	7	2	2

The number of factual questions in both the groups were almost equal. There were no conceptual questions in the English-only (control) group while there were seven conceptual questions in the Tamil+English (experimental) group. The number of procedural questions in the English-only and the Tamil+English group were one and two respectively. There were no metacognitive questions in the English-only group while there were two such questions in the Tamil+English group.

There were no follow-up questions in the English-only (control) group and there were five follow-up questions in the Tamil+English (experimental) group.

The questions across the four knowledge domains asked by the students in the experimental group are classified based on the language (Tamil or English) that was used to ask the question and the results are summarized in Table 6.

Table 6: Classification of questions across the four knowledge domains from the Tamil+English (experimental) group based on the language

Domain	English	Tamil
Factual	2	2
Conceptual	1	6
Procedural	0	2
Metacognitive	0	2

There were equal number of factual questions in Tamil and English. Six conceptual questions were asked in Tamil while one was asked in English. All procedural and metacognitive questions were asked in Tamil.

5 DISCUSSION

5.1 Interpretation of Results

Our study tried to find if using the native language (Tamil) along with English for teaching an upper-level CS course like the operating systems had any effect on students' learning of programming when compared to using only English. We measured the student learning in terms of gain scores and post-test scores.

The difference between the two groups with respect to the gain score is not statistically significant (see Section 4.2). This shows that teaching programming using Tamil and English is no different than teaching programming using only English. Also, we were unable to

measure the difference between the two groups with respect to the post-test score (with the pre-test score as a covariate) using ANCOVA since one of the ANCOVA assumptions (i.e., homogeneity of regression slopes) was not satisfied. Therefore, we need more quantitative experiments in this area to better understand the impact of the native language on students' understanding of upper-level CS concepts.

Our finding on the effect of native language on student learning in an operating systems course matches the findings of Soosai Raj et. al. [20] which found that the use of the native language (Tamil) did not have a significant positive impact on student learning in a data structures course. We consider our findings to be a validation of the previous results from an introductory programming course to an upper-level CS course.

On the other hand, our findings contradict the findings of Pal and Iyer [13] which suggests that the native language (Hindi) had a significant positive effect on students who did their schooling in a Hindi-medium school. The main reason for this difference might be that in our study even though the native language of all students was Tamil, the majority of students studied in an English-medium school during their K-12. On the other hand, 50% of students in Pal and Iyer's study were from a Hindi-medium background.

Based on the classroom observations we conducted as part of our study, we did a frequency count of the questions that were asked in the classroom, during our intervention, in both the groups. We found that giving students the freedom to speak using either their native language (Tamil) or English increased the classroom interaction that happened during the lectures (measured using the questions that were asked by the students in-class) in the Tamil+English (experimental) group when compared to the English-only (control) group (see Table 4).

We classified the questions along four knowledge dimensions based on the revised Bloom's taxonomy [2] and found that the students in the Tamil+English (experimental) group asked more questions about conceptual and procedural knowledge while the students in the English-only (control) group asked questions mainly about factual knowledge. We also found that 80% of the questions that students asked in the Tamil+English group were in Tamil. The use of the students' native language also encouraged the students in the Tamil+English group to ask follow-up questions in their native language while such a trend of asking follow-up questions was not found in the English-only group. Our results on analyzing the student questions shows that the use of the native language had an impact on the number and the quality of questions that non-native English speakers ask in a CS classroom.

An alternative interpretation of our results is that the students in the Tamil+English (experimental) group are better students (based on the pre-test scores) when compared to the students in the English-only (control) group. Therefore, the questions that these students asked might not be due to the native language but instead due to their inherent curiosity. We argue that this might not be the case since classroom observations that were conducted before our intervention showed that the amount of interaction that happened in the two groups were minimal and similar across the two groups.

Our results on the classroom interaction matches those observed by Airey [1] while teaching physics using English and Swedish. For example, Airey found that English is a barrier for students to ask questions during a lecture because students do not want to be embarrassed before their peers by asking a question in English. We also found that students in the English-only (control) group asked a lot fewer questions than students in the Tamil+English (experimental) group, where students were free to use their native language, if needed.

5.2 Limitations and Future Work

The main limitation of our study is the use of *nonrandomized control group pre-test post-test design*. We used this design mainly because we wanted our intervention to be part of regular classes. Therefore, we weren't able to randomize students into two different groups but instead used the two pre-formed groups as our experimental and control groups. As a consequence, the two groups varied significantly with respect to their prior knowledge in OS before the start of our intervention. Although we have taken this pre-test difference into consideration in our analysis of results, we acknowledge that the results would have been more reliable if these initial differences didn't exist among these groups. To minimize the effects due to the initial differences among the two groups, as a part of our future work, we plan to conduct more controlled experiments with a *randomized control group pre-test post-test design* [5] to better understand the effects of the native language for learning upper-level CS classes. We plan to do this by conducting special topics courses (outside students' regular course schedules) where the instructor will have the flexibility to randomize students into two different groups.

Another limitation with our study is that we did not control for the students' English proficiency. This is important because if the students in the two groups differed significantly in their English skills, then this might be an important confounding factor in our experiments. As part of our future experiments we plan to control for the students' English skills by conducting tests for English proficiency prior to our intervention.

6 CONCLUSION

We found that teaching an upper-level CS course (i.e., operating systems) using a bilingual teaching methodology (in Tamil and English) is no different than teaching using only English with respect to student learning measured using pre- and post- tests. We also found that the native language had a positive impact on the classroom interaction, measured using the quantity and the quality of student questions that were asked during the class. We conclude that more studies should be conducted in bilingual CS education, across different CS courses and using different natural languages (e.g., Spanish) to truly understand and benefit from the role that native language plays in computer science education.

ACKNOWLEDGEMENTS

Our sincere thanks to Dr. Saswati Mukherjee, Ms. V. Ezhil Arasi and Mr. B. R. Yuvaraj for allowing us to work with their students and helping us conduct these experiments. We also thank all the students who took part in our experiments.

REFERENCES

[1] John Airey. 2009. *Science, language, and literacy: Case studies of learning in Swedish university physics.* Ph.D. Dissertation. Acta Universitatis Upsaliensis.

[2] Lorin W Anderson, David R Krathwohl, Peter W Airasian, Kathleen A Cruikshank, Richard E Mayer, Paul R Pintrich, James Raths, and Merlin C Wittrock. 2001. A taxonomy for learning, teaching, and assessing: A revision of BloomâĂŹs taxonomy of educational objectives, abridged edition. *White Plains, NY: Longman* (2001).

[3] Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau. 2015. *Operating Systems: Three Easy Pieces* (0.91 ed.). Arpaci-Dusseau Books.

[4] Russ Cox, M Frans Kaashoek, and Robert Morris. 2011. Xv6, a simple Unix-like teaching operating system. (2011).

[5] Dimiter M Dimitrov and Phillip D Rumrill Jr. 2003. Pretest-posttest designs and measurement of change. *Work* (2003).

[6] Jennifer R Fennema-Bloom. 2009. Code-scaffolding: A pedagogic code-switching technique for bilingual content instruction. *Journal of Education* (2009).

[7] Ofelia García and Claire E Sylvan. 2011. Pedagogies and practices in multilingual classrooms: Singularities in pluralities. *The Modern Language Journal* (2011).

[8] Gene V Glass and Kenneth D Hopkins. 1970. *Statistical methods in education and psychology.* Prentice-Hall Englewood Cliffs, NJ.

[9] Philip J Guo. 2018. Non-Native English Speakers Learning Computer Programming: Barriers, Desires, and Design Opportunities. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems.* ACM, 396.

[10] Brian W Kernighan and Dennis M Ritchie. 2006. The C Programming Language. (2006).

[11] Thomas Andrew KIRKPATRICK. 2011. Internationalization or Englishization: Medium of instruction in today's universities. (2011).

[12] T Murugavel. 2011. The Problems of Non-English Medium Engineering Students and Possible Solutions. (2011). http://worldlitonline.net/the-problems-of-non-english.pdf

[13] Yogendra Pal. 2016. *A Framework for Scaffolding to Teach Programming to Vernacular Medium Learners.* Ph.D. Dissertation. IIT, Bombay.

[14] Yogendra Pal and Sridhar Iyer. 2012. Comparison of English versus Hindi Medium Students for Programming Abilities Acquired through Video-Based Instruction. In *T4E.* IEEE.

[15] Yogendra Pal and Sridhar Iyer. 2015. Classroom Versus Screencast for Native Language Learners: Effect of Medium of Instruction on Knowledge of Programming. In *ITiCSE.* ACM.

[16] Yogendra Pal and Sridhar Iyer. 2015. Effect of medium of instruction on programming ability acquired through screencast. In *LaTiCE.* IEEE.

[17] Margaret Probyn. 2001. Teachers voices: Teachers reflections on learning and teaching through the medium of English as an additional language in South Africa. *Bilingual Education and Bilingualism* (2001).

[18] Margaret Probyn. 2005. Learning science through two languages in South Africa. In *The 4th International Symposium on Bilingualism, Somerville, MA.*

[19] Adalbert Gerald Soosai Raj, Kasama Ketsuriyonk, Jignesh M Patel, and Richard Halverson. 2017. What Do Students Feel about Learning Programming Using Both English and Their Native Language?. In *2017 International Conference on Learning and Teaching in Computing and Engineering (LaTICE).* IEEE, 1–8.

[20] Adalbert Gerald Soosai Raj, Kasama Ketsuriyonk, Jignesh M Patel, and Richard Halverson. 2018. Does Native Language Play a Role in Learning a Programming Language?. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education.* ACM, 417–422.